НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету

Обчислювальної техніки
(повна назва кафедри

До захисту допущено
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)
“___”_____2019р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050103 « Програмна інженерія »**
(код і назва)

на тему: Веб-сервіс для провайдера статистичних даних

Виконав (-ла): студент (-ка) __4__ курсу, групи __ІП-54_____
(шифр групи)

_____Щепець Артем Анатолійович_____
(прізвище, ім'я, по батькові) (підпис)

Керівник Доц. каф. ОТ, к.т.н. Доц. Бондак А.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Нормоконтроль д.т.н. Сімоненко В.П.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент Доц. каф. АСОІУ, к.т.н. Доц. Мука І.П.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО"

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету

Обчислювальної техніки

Освітньо-кваліфікаційний рівень **бакалавр**

Напрям підготовки **6.050103 « Програмна інженерія »**

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(прізвище ініціали) (підпис)
"___" _____2019 р.

ЗАВДАННЯ
на бакалаврську дипломну роботу студента

Щепця Артема Анатолійовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Веб-сервіс для провайдера статистичних даних*
керівник проекту (роботи)_ Болдак Андрій Олександрович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "23"04 2019 року №1180-С
2. Термін здачі студентом закінченого проекту (роботи)___ 20 травня 2019р.
3. Вихідні дані до проекту (роботи) *технічна документація, теоретичні та статистичні дані, патенти на винахід*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Опис предметної області, дослідження методики побудови електронного підручника на базі гіпертекстової технології, програма забезпечення гіпертекстових технологій.
5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)
концепт-діаграма, схема прецендентів, блок-схема алгоритма.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
/.	*Затвердження теми роботи*	10.12.2018-15.12.2018	
2.	*Вивчення та аналіз завдання*	15.12.2018-15.03.2019	
3.	*Розробка архітектури та загальної структури системи*	15.03.2019 – 25.03.2019	
4.	*Програмна реалізація системи*	5.04. 2019-15.04.2019	
5.	*Оформлення пояснювальної записки*	15.04. 2019-20.05.2019	
6.	*Передзахист*	29.05.2019	
7.	*Захист*	20.06.2019	

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

Анотація

В бакалаврській *дипломній* роботі реалізовано веб-сервіс для провайдера статистичних даних, призначеного для комфортного та швидкого обміну та зберіганню великих обсягів даних, який базується на принципі відкритих даних (open-data) та моделі даних DDF.

Програма надає оптимізований пошук даних, дозволяє формувати та завантажувати вибірки даних певних критеріїв, доповнювати інформацію метаданими. Програмний продукт був створений на мові Java у візуальному середовищі IntelliJ IDEA.

Для візуалізації, вводу та отримання інформації у графічному інтерфейсі використовується фреймворк ReactJS та мова програмування JavaScript.

Аннотация

В бакалаврской дипломной работе реализовано веб-сервис провайдера статистических данных, предназначенного для комфортного и быстрого обмена и хранения больших объемов данных, основанный на принципе открытых данных (open-data) и модели данных DDF.

Программа предоставляет оптимизированный поиск данных, позволяет формировать и загружать выборки данных определенных критериев, дополнять информацию метаданными. Программный продукт был создан на языке Java в визуальной среде IntelliJ IDEA.

Для визуализации, ввода и получения информации в графическом интерфейсе используется фреймворк ReactJS и язык програмирования JavaScript.

Annotation

In the Bachelor thesis, a web-based service is provided for the provider of statistical data intended for the systematization, exchange and storage of large volumes of data, based on the principle of open data (open-data) and DDF data models.

The program provides optimized data retrieval, allows you to generate and download sample data of certain criteria, to supplement information with metadata. The software product was created in the Java language in the visual environment of IntelliJ IDEA.

To visualize, enter and retrieve information in the graphical interface, the framework ReactJS and JavaScript language are used.

ЗМІСТ

					ІАЛЦ.467100.002 ТЗ		
мн.		№ докум.	Підпис	Дата			
Розробн.		Щепець А.А.			Веб-сервіс для провайдера статистичних даних. Пояснювальна записка	Літ.	Арк. 1 / Аркушів 3
Перевір.		Болдак А.О.					
Н. Контр.		Сімоненко В.П.				НТУУ "КПІ",	
Затверд.		Стіренко С.Г.					

ПЕРЕЛІК СКОРОЧЕНЬ

ОС	Операційна система
БД	База даних
SQL	(Structured Query Language) Мова структурованих запитів
SPA	(Single Page Application) Односторінковий додаток
HTML	(Hypertext Markup Language) Мова розмітки гіпертекстових документів
CSS	(Cascading Style Sheets) Каскадні таблиці стилів
JS	JavaScript
HTTP	(Hyper Text Transfer Protocol) Протокол передачі гіпертексту
DOM	(Document Object Model) Об’єктна модель документу
URL	(Uniform Resource Locator) Єдиний вказівник на ресурс
API	(Application programming interface) Програмний інтерфейс додатку
IDE	(Integrated Development Environment) Інтегроване середовище розробки
CLR	(Common Language Runtime) Загальномовне виконуване середовище
JIT	(Just-in-time compilation) Механізм компіляції «на льоту»
CLI	(Command Line Interface) Інтерфейс командного рядку
REST	(Representational State Transfer) Передача репрезентативного стану
JSON	(JavaScript Object Notation) Запис об’єктів JavaScript
XML	(Extensible Markup Language) Розширювана мова розмітки
WPF	(Windows Presentation Foundation) Система для побудови клієнтських, десктопних додатків на ОС Windows
WCF	(Windows Communication Foundation) Програмний фреймворк для комунікації між додатками, що входить в .NET Framework

UWP	(UWP) Універсальна платформа Windows
DQL	(Data Query Language) Мова запитів до даних
DDL	(Data Definition Language) Мова опису даних
DCL	(Data Control Language) Мова контролю даних
DML	(Data Manipulation Language) Мова маніпулювання даними
ACID	(Atomicity, Consistency, Isolation, Durability) Перелік властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, консистентність, ізольованість, довговічність
СУБД	Система управління базою даних
MVC	(Model-View-Controller) Модель-вигляд-контолер
CORS	(Cross-Origin Resource Sharing) Технологія спільного використання ресурсів з різних джерел
ORM	(Object Relation Model) Об'єктно-реляційна проекція
CRUD	(Create, read, update, delete) Базові операції створення, зчитування, оновлення та видалення
LINQ	(Language Integrated Query) Технологія інтегрованих в мову запитів
JWT	(JSON Web Token) Веб-токен в форматі JSON
RPC	(Remote procedure call) Виклик віддалених процедур

ВСТУП

В сучасних умовах розвитку суспільства значно зріс інтерес до статистики як науки та її широкому застосуванню в практичній діяльності. Вже ніхто сьогодні не зможе заперечити значення та недооцінити роль статистики у сучасному житті.

Статистичні дані сприяють формуванню адекватного представлення про нинішнє економічне, екологічне, демографічне та багатьох інших положень справ у країні та світі в цілому. Саме завдяки статистиці у разі виявлення яких-небудь відхилень або невідповідностей стає можливим своєчасно вжити ряд коригуючих дій та поліпшити ситуацію.

Однак, є ряд проблем, які роблять дуже важким систематизацію, зберігання та пошук необхідних статистичних даних. Перша і найголовніша проблема - немає певного одного зразку зберігання та передачі статистичних даних. Статистика надходить з багатьох джерел та зберігається у різному вигляді. Це уповільнює процеси обміну та пошуку статистичних даних.

Метою моєї дипломної роботи є створення спеціалізованого веб-сервісу за концепцією відкритих даних, який допоможе вирішити сучасні проблеми статистичних провайдерів, такі як прискорення процесів обробки даних та доступ до інформації з будь-якого місця, де є підключення до мережі Інтернет. Моя система вирішить проблему з певним зразком зберігання даних. Усі дані будуть надходити як до сервера, так і до клієнта у вигляді DDF моделі даних. Користувач зможе оцінити та проаналізувати графічне зображення статистичних даних у декількох форматах. Також, веб-сервіс буде підтримувати функцію Синонімів, яка спростить пошук по ключовим словам та Метадані, завдяки яким інформація для користувача буде подаватися максимально розгорнуто та детально.

У даній дипломній роботі описуються етапи проектування, розробки та використання сервісу для управління статистичними даними.

Для успішної реалізації даного сервісу потрібно вирішити декілька завдань, а саме:

- написання функціональної частини додатку (бекенд);
- розробка веб-інтерфейсу для користувачів (фронтенд);
- проектування структури бази даних.

Провівши аналіз засобів розробки додатків та існуючих мов програмування, я обрав фреймворк Spring та мову програмування Java для реалізації бекенду веб-сервіса. Для написання фронтенду проекта мною був обраний фреймворк ReactJS та мова програмування JavaScript. В якості СУБД я обрав MySQL.

РОЗДІЛ 1
ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ, АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА ФОРМУВАННЯ ВИМОГ

В моєму дипломному проекті створюється спеціалізований веб-сервіс на основі концепції відкритих даних, який вирішує основні проблеми провайдерів статистичних даних, таких як оптимізація процесів пошуку, збереження, видалення та редагування даних, в зв’язку з переходу від застарілих засобів ведення обліку на сучасні онлайн-додатки – зведення всіх статистичних даних до одного певного формату, надання більш детальної інформації та створення максимально зручного і зрозумілого інтерфейсу для користувача.

Управління статистичними даними - це дуже важливий вид фахової аналітичної діяльності, яка націлена на досягнення мети через розумну організацію функцій, методів та принципів управління збору, представлення, аналізу та розумної інтерпретації даних.

Статистика являється дуже строгим науковим методом, що дозволяє зрозуміти дані, дійти до їх суті. Наприклад, у дослідженні може вимірюватися вага 1000 пацієнтів. Це уже достатньо велика кількість спостережень, і просто глянувши на дані неможливо отримати швидке та інформативне уявлення. Однак статистика може дати миттєву загальну картину даних — на основі доступної для сприйняття візуалізації або числового узагальнення — незалежно від кількості спостережень чи одиниць даних.

Дані представляють собою спосіб представлення, зберігання, елементарних операцій обробки та аналізу інформації. Зазвичай дані є вхідною інформацією для інформаційного процесу.

Дані, як речовину чи енергію, можна збирати, обробляти, зберігати, змінювати форму їх представлення. Їх можна створювати, знищувати,

багаторазово використовувати. Найголовнішою особливістю даних на сьогоднішній день є те, що їх стає надзвичайно багато. При масовому та масштабному застосуванні ЕОМ виникла величезна кількість джерел інформації. Для прикладу можна взяти обсяг даних у всесвітній мережі Інтернет, що збільшується кожної секунди з великою швидкістю.

Головним поняттям управління даними є структура типу "файл", що представляє собою множину однотипних елементів, тобто записів. Також файл займає певну ділянку на носії пам'яті та характеризується ім'ям, типом та іншими атрибутами. В свою чергу запис - це структура, яка складається з полів (мінімальної структури даних).

Основними етапами життєвого циклу даних є створення (виникнення), зберігання, використання та знищення. Знищення, з точки зору життєвого циклу даних, не представляє ніякого інтересу, оскільки результатом видалення є втрата інформативності даних. Етап використання даних включає три підпункти:

- пошук даних;
- обробку даних;
- аналіз даних.

Результатом використання даних є інформація.

Існує декілька методів збору, які необхідні саме для аналізу даних:

- збір з облікових систем;
- збір непрямих даних;
- збір відкритих джерел;
- проведення незалежних маркетингових досліджень і аналогічних заходів щодо збору даних;
- збір внутрішніх даних.

Так як мій дипломний проект буде створений на основі концепції відкритих даних, зупинимося більш детально на відкритих джерелах інформації.

Відкриті дані (або open data) — це такий підхід до зберігання інформації, згідно з яким набір даних має бути у вільному доступі, вільно використовуватися та розповсюджуватися. При цьому такі дані можуть використовувати як фізичні особи(звичайні люди) та неприбуткові організації, так і комерційні установи. Використання відкритих даних із комерційною метою не є забороненим та навіть заохочується.

До відкритих даних мають застосовуватися в обов'язковому порядку такі параметри:

• підтримка сервісу з даними цілодобово, його постійна доступність;

• відсутність будь-яких обмежень на рівнях доступу (паролів, які перешкоджають перегляду та інше);

• безкоштовність та анонімність надання даних та їх використання;

• можливість завантажувати у популярних та розповсюджених форматах, що є універсальними (.doc, .excel, .pdf, .jpg, csv та інші);

• доступне та зрозуміле API задля поширення та використання на інших сторонніх платформах;

• отримання даних виключно з офіційних та перевірених ресурсів.

Серед джерел, у яких слід шукати відкриті дані:

• онлайн-представництва установ та їхні електронні сховища даних. Там оприлюднюються дані від міністерств. відомств, державних органів та установ на рівні міст, які попередньо ретельно перевіряються. До числа таких ресурсів належить data.gov.ua),

• великі корпорації, комерційні підприємства та організації,

• соціальні мережі та платформи.

Дані, які надає держава є одними з ключових інтересів для суспільства. Численні некомерційні організації та деякі активісти домагаються відкритості усієї державної інформації у формі, що може підлягати машинній обробці. Багато національних урядів в рамках стратегій «відкритої держави»(open-gov) створили веб-сервіси для поширення частини даних, які обробляються у секторі державного управління.

Деякі відомі сайти ву стратегії open-data:

- data.gov (портал відкритих даних США),
- science.gov (портал відкритих даних США, присвячений науці та технологіям),
- data.gov.uk (портал відкритих даних Англії),
- data.gov.in, (портал відкритих даних Індії),
- data.gov.ua (портал відкритих даних України).

Після пошуку і збору дані перетворюються у єдиний формат, наприклад, таблицю Excel, текстовий файл, чи компонент довільної БД. Однією із важливих дій при цьому є визначення способу представлення даних. Зазвичай, це один з наступних видів - число, рядок, дата, булева логічна змінна (так/ні). Формалізувати (визначити спосіб представлення) деякі дані буває дуже легко - наприклад, об'єм продажів у гривнях - це конкретне число. Але, дуже часто, виникає ситуація, коли представлення чинника є зовсім невідомим. Зазвичай, такі проблеми виникають з якісними характеристиками. Наприклад, відомо, що на обсяги продажів впливає якість товару (як для продажу побутової техніки чи одягу).

Дані повинні бути уніфікованими - одні й ті ж самі дані скрізь повинні описуватись однаково. Часто при видобуванні знань основну увагу приділяють механізмам аналізу даних, не враховуючи важливість попередньої обробки та очищення даних. Очевидно, що некоректні початкові дані призводять до некоректного аналізу та висновків. Зазначимо, що

зазвичай, джерелом інформації для аналітичних систем є сховище даних, в якому акумулюються відомості з різних джерел, тому гострота цієї проблеми суттєво зростає.

Проблему з уніфікацією даних легко вирішить модель даних DDF. Модель даних DDF застосовується для того, щоб визначити набори даних (datasets). Набір даних (dataset) - це скелет узгоджених та зв'язаних між собою даних які складаються з окремих елементів, але якими можно керувати як однією одиницею даних. Кожен набір даних містить в собі 3 колекцій даних: Концепти (Concepts), Метадані (Metadata), Датапоінти (Datapoints).

1. Датапоінти містять в собі багатоаспектні дані.
   Наприклад: *Чоловіче населення Німеччини у 2015 році становило 293956.*
2. Метадані містять додаткову інформацію.
   Наприклад: *Джерело усіх даних про населення у 2015 році взято з центрального бюро статистики та має точність в 96%.*
3. Концепти зберігають в собі інформацію про змінні в наборі даних (тобто про заголовки стовпців у табулярному форматі)
   Наприклад: *Населення вимірюється у кількості людей та повинно будти більше нуля.*

Набір даних *мусить* мати Концепти та *може* мати Датапоінти та Метадані.

Кожен елемент цих колекцій складається з пар “ключ-значення” (key-value). Кожна така пара є унікальною по всьому набору даних. (Більш детально про модель DDF у розділі 2).

Основна мета моєї дипломної роботи – це допомогти в управлінні статистичними даними, відповідати змінам ринку, створювати та поглиблювати і підтримувати конкурентні переваги. Для реалізації цього

завдання потрібно побудувати таку інформаційно-технологічну систему, яка відповідатиме таким критеріям:

- загальнодоступність – користувач може отримати доступ до ресурсів веб-сервісу з будь-якого місця та у будь-який час;
- система повинна бути доступною для великої кількості користувачів одночасно;
- масштабованість прикладних програм.

1.1 Аналіз сервісу «data.worldbank.org»

Для формування уявлення про систему, яку я реалізовую, розглянемо готові рішення, а саме сервіси відкритих даних (open data services).

Сервіс, який збирає всю важливу фінансову\екологічну\демографічну статистику з усього світу та надає її у вільний доступ. Розроблений Світовим Банком.

Світовий банк є однією з найбільших у світі організацій, що надають допомогу країнам з метою їх розвитку. Основною характеристикою сервісу є його достовірність (більшість даних надавалися урядами країн та великими бізнес-корпораціями) та величезні об’єми даних: більш ніж 30000 різноманітних наборів даних.

Сервіс має закриту архітектуру, усі операції з БД здійснюється за допомогою API. Веб-сервіс має графічний інтерфейс, однак користувач може лише переглядати дані без додаткових можливостей скачування та завантаження файлів (вони можливі лише через API).

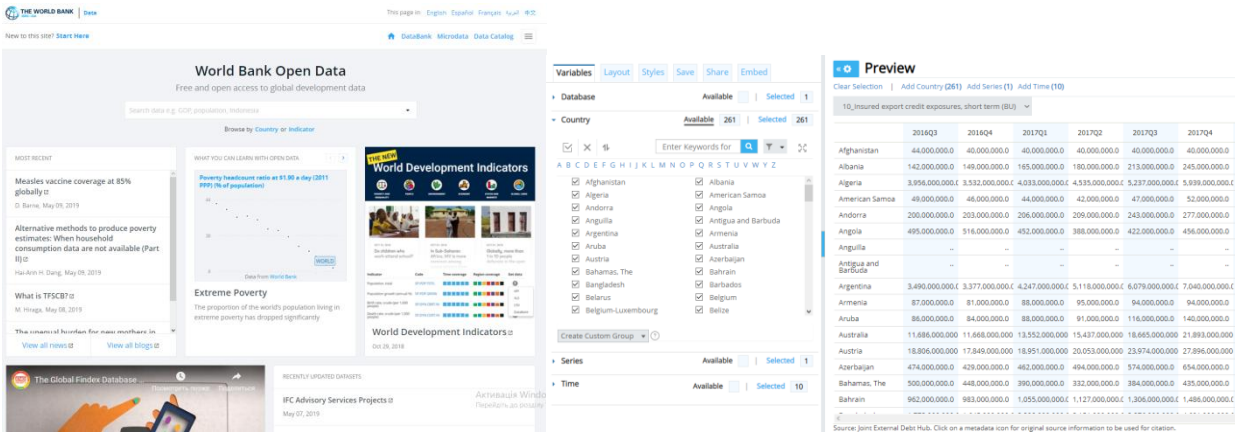

Рис. 1.1а та рис. 1.1б - зовнішній вигляд сервісу WorldBank

Переваги системи:

- Доступ до системи здійснюється з будь-якого місця та девайсу, де є підключення до мережі Інтернет.
- Має швидкий і зручний пошук завдяки структурованим категоріям даних та підтримці пошуку ключових слів по синонімам.
- Максимальна достовірність статистичних даних.
- Постійне оновлення та поповнення БД новими статистичними даними – кожен день додається понад 30 нових датасетів.
- Містить велику кількість інструментів для більш точного та детального розгляду статистичних інструментів.
- Мають метадані, які дозволяють отримати додаткову інформацію користувачу.

Недоліки системи:

- Складний та громіздкий графічний інтерфейс, який потребує деякого часу на його вивчення для зручної роботи користувача.
- Закрита архітектура проекту.
- Всі основні дії проходять через API, а не через графічний інтерфейс.
- Неможливість завантаження статистичних даних без знання мов програмування (здійснюється через API).

1.2 Аналіз сервісу «Citi Bike NYC»

Citi Bike NYC – це ще один приклад успішного сервісу провайдера статистичних даних, але більш вузький, який спеціалізується лише на статистиці велосипедистів.

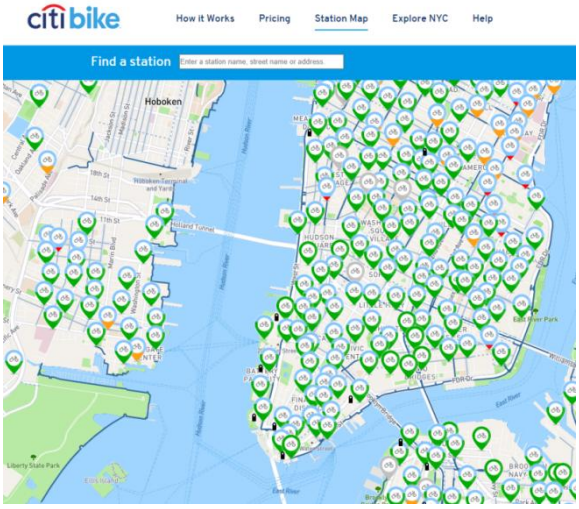

Рис. 1.2 - зовнішній вигляд сервісу CitiBike

Плюси сервіса:

- Інтерактивна мапа, за допомогою якої візуалізуються статистичні дані.
- Збір статистики у реальному часі. Веб-сервіс публікує у реальному часі системну статистику у спеціальний сервіс, до якого можна під'єднатися за допомогою API.

Мінуси сервіса:

- Вузька спеціалізація. Збір статистики ведеться лише по велосипедистам та лише у місті Нью-Йорк і тільки на тих велосипедах, які орендуються, оминаючи власників особистих велосипедів.
- Нехтування програмою деякими показниками (такими, як, наприклад, поїздкою, яка тривала менше 2-3 хвилин) та взяття швидкості як константи (уся швидкість записується як середнє значення по всім велосипедам - 7.5 мілі\год). Усе це робить статистичні дані менш точними та детальними.

- Відсутність графічного інтерфейсу у веб-сервісу для тих, хто хоче скористатися їхніми статистичними даними. Всі дії з сервісом відбуваються через SQL запити, що робить неможливим використання сервісу для звичайних користувачів.

1.3 Аналіз сервісу «data.gov.ua»

Сервіс, створений нашим урядом в рамках стратегій «відкритої держави». Створений для поширення частини даних, що обробляються в секторі державного управління.

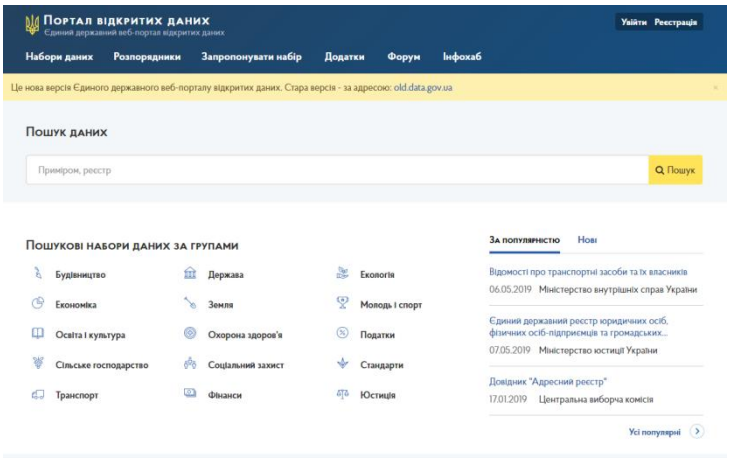

Рис. 1.3 - зовнішній вигляд сервісу Data.gov.ua

Плюси системи:

- Можливість завантажувати свої набори даних до сервісу, запропонувавши їх на відповідній сторінці.
- Постійне оновлення наборів даних
- Доступ з будь-якого девайсу з підключенням до Інтернету

Мінуси системи:

- Дуже мала кількість наборів даних.
- Незрозумілий та недопрацьований графічний інтерфейс.

- При завантаженні будь-якого набору даних користувач отримає xml файл без будь-якої інструкції як з ним взаємодіяти. Можливості завантаження файлів одразу у табличних форматах немає.
- Застарілий та не оптимізований пошук.

1.6 Порівняння існуючих програмних рішень

В результаті проведеного аналізу мною сформовано порівняльну таблицю 1.1. на якій представлено переваги та недоліки розглянутих програмних продуктів.

Таблиця 1.1 – Порівняння розглянутих рішень

Критерій	data.worldbank.org	Citi Bike NYC	data.gov.ua
Можливість безкоштовно використовувати весь функціонал	+	-	+
Ціна	безкоштовно	дод. функц. за 30$/місяць	безкоштовно
Модульність	+	-	-
Кросплатформність	+	+	+
Функціональний та комфортний інтерфейс	-	-	-
Захищеність даних	+	+	+
Оптимізований та зручний пошук	+	+	-
Зручність завантаження даних	-	-	-
Відкрита архітектура	-	-	-

Проаналізувавши попередні продукти можна підсумувати, що основними умовами майбутнього проекту є функціональність та зручність його графічного інтерфейсу та пошуку, можливість завантаження даних, а також відкрита архітектура проекту.

Також буде підтримуватися тільки україномовний варіант графічного інтерфейсу, однак у ході розширення планується додати інші локалізації та нові функції. Ще однією перевагою є відсутність примусового оновлення частин системи.

1.7 Формування вимог до майбутньої системи

Після того як я ознайомився з готовими рішеннями на ринку, мною було вирішено створити систему, яка буде задовольняти наступні критерії:

а) повинен бути функціональний та зрозумілий інтерфейс;

б) система повинна мати відкриту архітектуру;

в) мати можливість збереження даних з різних джерел;

г) для забезпечення конфіденційності повинна бути організована функціональність для аутентифікації користувачів;

ґ) для різних користувачів має бути організована окрема специфічна функціональність;

д) система повинна мати оптимізований та комфортний пошук;

Тепер потрібно визначитись з архітектурою, згідно якої буде проектуватися система. Для даної системи я обрав трирівневу архітектуру, яка буде складатися з клієнтської частини (веб-браузер), серверу та серверу бази даних.

Для побудови даного веб-сервісу, перерахуємо ряд вимог, які потрібно вирішити:

- можливість реєстрації в системі, для подальшої аутентифікації користувачів;

- функціональність авторизованого входу до системи, як для адміністраторів, так і для користувачів даної системи;
- можливість створення, редагування та видалення наборів даних;
- можливість завантаження файлів у систему;
- можливість графічного зображення статистичних даних у вигляді графіків та стовпчикових діаграмах.
- можливість пошуку даних по ключовим словам та словосполученням, зрозуміле відображення результатів пошуку.
- Можливість пошуку даних за схожими словами-синонімами.
- можливість надання інформації за певними категоріями, їхнє створення та видалення з БД.
- можливість відображення історії переглядів користувача.
- можливість надання додаткової інформації для користувача за допомогою метаданих.

Майбутня функціональність нашої системи визначена, перейдемо до засобів реалізації задумки.

Висновки до розділу

1. Було створене підґрунтя для подальшої розробки програмної системи. Також, був здійснений пошук та аналіз потрібної інформації щодо розроблення системи.
2. Виконано аналіз трьох існуючих програмних рішень для того, щоб не повторювати помилки, які є в даних системах та для того, щоб перейняти все найкраще, що в них є.
3. Порівняно існуючі системи та сформовано основні умови для майбутнього проекту.
4. Сформовано вимоги до системи, за якими і буде вестися розробка додатку.

РОЗДІЛ 2

ВИБІР АРХІТЕКТУРИ ТА ПРОЕКТУВАННЯ СИСТЕМИ STATSERV

У процесі розробки предмету виконання моєї дипломної роботи було прийнято рішення застосувати підхід з трирівневою архітектурою. Розглянемо детальніше цей підхід.

Трирівнева архітектура - це застосування більш загальної багаторівневої моделі. Вона заснована на середовищі клієнт-сервер. Логічна архітектура системи поділена на три рівні або шари:

1. Шар презентацій (графічний компонент), який являє собою перший рівень, а саме представлення застосунку для кінцевого користувача. Цей рівень не повинен мати зв’язків з базою даних та бути навантаженим будь-якою бізнес-логікою.

Частіш за все на перший рівень виносять найпростішу бізнес-логіку, наприклад:

- інтерфейс авторизації користувачів;
- перевірка введених значень на коректність та відповідність формату;
- алгоритми шифрування даних;
- прості операції з даними, які були вже попередньо закешовані чи завантажені на комп'ютер юзера;

2. Шар обробки (серверна частина).

На цьому рівні розташовується сервер застосунків. Також тут розташовується більша частина бізнес-логіки.

3. Рівень доступу до даних (сервер БД).

Цей рівень призначений для розташування серверу БД, який забезпечує збереження даних. Такий функціонал може виконувати реляційна або об'єктно-орієнтована СУБД. Рівень доступу до даних повинен забезпечувати API рівня застосування, який надає методи управління зберігаємими даними, не піддаючи чи не створюючи залежності від механізмів зберігання даних. Завдяки цьому можна змінювати структуру програми так, що звичайний користувач не буде знати, що ці зміни взагалі буди внесені.

Переваги цієї архітектури у порівнянні з клієнт-серверною архітектурою:

- набагато вищий рівень надійності;
- легка масштабованість;
- високий рівень безпеки;
- незначні вимоги до швидкості передачі даних між користувачем та сервером;
- налаштованість системи – оскільки всі рівні ізольовані один від одного, це дозволяє швидко переналаштовувати систему при виникненні проблем чи збоїв на одному з рівнів;
- дуже незначні вимоги до технічних характеристик та продуктивності гаджетів (комп’ютерів, мобільних телефонів).

2.1 Огляд клієнтської частини додатку та інструментів (Фронтенд)

Розпочнемо огляд архітектури та інструментів використаних в клієнтській частині додатку. Основними її завданнями є: реагування на дії користувача, надання зручного ы швидкого графічного інтерфейсу, комунікація з сервером додатку та обробка наданих ним відповідей, а також коректна обробка помилок, які можуть виникнути під час взаємодії клієнта з серверною частиною додатку.

З точки зору вибору архітектурного підходу для розробки клієнтської частини було вирішено імплементувати її, як односторінковий додаток (SPA). Односторінковий додаток – це веб-додаток, який взаємодіє з користувачем, динамічно переписуючи поточну сторінку, а не завантажуючи нову. Цей підхід сприятливо впливає на плавність роботи додатку, хоча завантаження першої сторінки може зайняти трішки більше часу, ніж у випадку з багатосторінковими додатками. Даний підхід до роботи з веб-сайтом дозволяє йому бути більш динамічним та поводитись схожим чином до десктопних аналогів. Вперше ідея була реалізована Стюартом Моррісом в 2002 році.

Проте, зазвичай, SPA не пишуться з нуля. Так як логіка підвантаження та маніпулювання частинами DOM є нетривіальною, то більшість веб-розробників надає перевагу використанню існуючих фреймворків. Найбільш популярними з них є Vue.JS, React, Angular, Ember.js ExtJS та KnockoutJS.

2.1.1. Фреймворк ReactJS

Для забезпечення всіх вищезгаданих властивостей було прийняте рішення використати фреймворк ReactJS для розробки клієнтської частини.

ReactJS – це фреймворк, створений для побудови клієнтських додатків з застосуванням HTML, CSS та JavaScript. Основним архітектурним принципом даного фреймворку є компонентна орієнтованість. Це означає, що додаток розбитий на певну сукупність компонент. Вони мають однакову структуру і це надає додатку певної шаблонності. У порівнянні з іншими аналогами було виділено наступні переваги, які виділяють фреймворк серед інших:

- React - це не тільки набір інструментів, а й паттернів, що полегшують розуміння, написання та підтримку коду.

- Велика кількість доповнень, бібліотек та плагінів, які розширюють можливості платформи.
- Підтримка різноманітних видів тестів.
- Активно підтримується та розвивається.

Головним інтегрованим середовищем для розробки клієнтської частини додатку було обрано JetBrains WebStorm.

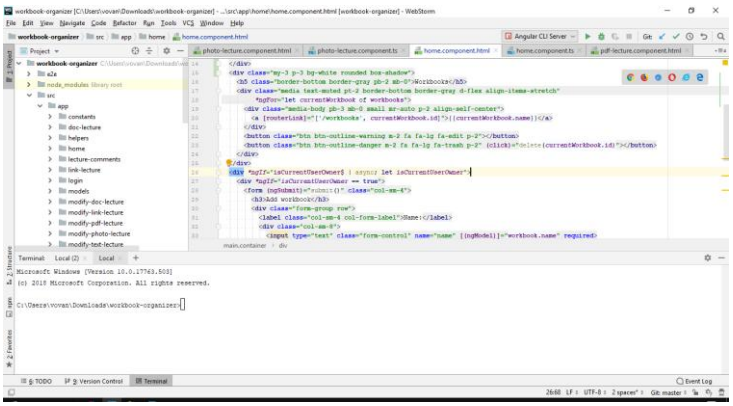

Рис.2.1 – Зовнішній вигляд IDE WebStorm

Серед основних переваг WebStorm можна виділити:

- Приємний та інтуїтивно-зрозумілий користувацький інтерфейс.
- Хороша підтримка механізму автодоповнення.
- Велика кількість функціоналу спрямованого на підвищення якості коду та рефакторинг.
- Зручна навігація по файлам та типам, швидкий пошук по всьому проекту.
- Підтримка систем контролю версій.
- Інтегрований термінал у вікно IDE.
- Можливість відлагодження клієнтського додатку.
- Наявність програми JetBrains для студентів, що дозволяє безкоштовно використовувати даний додаток в навчальних цілях.

Переваги наведені вище роблять WebStorm одним з найпопулярніших IDE серед веб-розробників. В порівнянні з такими конкурентами як Visual Studio, Visual Studio Code, Notepad++, Sublime Text продукт спеціалізується здебільшого на клієнтській частині веб-додатків. Цей фокус дозволяє ідеально оптимізувати додаток під ці потреби.

2.1.2 Мова програмування JavaScript

Для створення односторінкового веб-застосунку та сценаріїв веб-сторінок я обрав мову програмування JavaScript.

JavaScript (JS) — це динамічна ООП мова програмування, яка надає можливість на стороні клієнта (пристрої користувача: телефоні\планшеті\компьютері) взаємодіяти з самим користувачем та керувати браузером, асинхронно обмінюватися даними з сервером, а також змінювати структуру і зовнішній вигляд сторінки.

JavaScript класифікують як прототипну скриптовану мову програмування з динамічними типами даних. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та трохи функціональну) та деякі відповідні архітектурні властивості, наприклад: динамічна і слабка типізація даних, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

JavaScript має багато властивостей ООП мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов об’єктно орієнтованого програмування. Крім того, JavaScript має низку властивостей, які притаманні функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання (closures) — що додає мові додаткову гнучкість.

2.2 Огляд серверної частини додатку та інструментів (Бекенд)

Почнемо огляд архітектури серверної частини веб-сервісу. Її основним призначенням є обробка запитів, реалізація бізнес-логіки додатку, коректне збереження та валідація даних, взаємодія з БД.

Серверна частина додатку – це веб-додаток, який реалізує REST-архітектуру, розроблений на платформі Spring Framework з використанням модулів Spring MVC, Spring Data, Spring Security та мови програмування Java.

У ході проектування дипломної роботи мною було обрано REST-архітектуру, бо:

- У порівнянні з REST, архітектура аналога SOAP масивна та важка з точки зору машинної обробки. Тому REST працює набагато швидше.
- Відповідь від серверу за допомогою REST може бути представлена у різних форматах (в той час як його аналог SOAP прив’язаний тільки до XML).

Розглянемо дану архітектуру більш детально.

REST – це стиль архітектури програмного забезпечення, який як правило застосовується для побудови веб-служб. Термін REST був введений Роєм Філдінгом, який був одним з авторів HTTP-протокола. Системи, які підтримують REST архітектуру, називаються RESTful - системами.

В RESTful-системі усі одиниці інформації інформації визначаються глобальним ідентифікатором, а саме URL. Кожний ідентифікатор URL в свою чергу має строго заданий формат.

Нижче більш детально про можливості REST:

- Відсутність додатковий внутрішніх прошарків означає передачу даних в тому самому вигляді, що і самі дані. Тобто, ми не завертаємо дані у XML, не використовуємо AMF і т.д. Лише віддаємо свої дані.
- Кожна одиниця інформації однозначно визначається URL - це значить, що URL – це перший ключ до одиниці даних. Тобто, наприклад, 3-тя книга з книжної полки буде мати вигляд .../book/3, а 35 сторінка в цій самій книзі .../book/3/page/35.

Як відбувається управління інформацією сервіса - воно цілком і повністю базується на протоколі передачі даних. Найбільш поширений протокол, звісно, HTTP. Для HTTP дії над даними задаються за допомогою методів: GET(отримати), PUT(додати, замінити), POST (додати, замінити, видалити), DELETE (видалити). Звідси витікає, що CRUD-операції можуть виконуватися як з усіми 4-ма методами (GET/PUT/POST/DELETE), так і тільки за допомогою методів GET та POST.

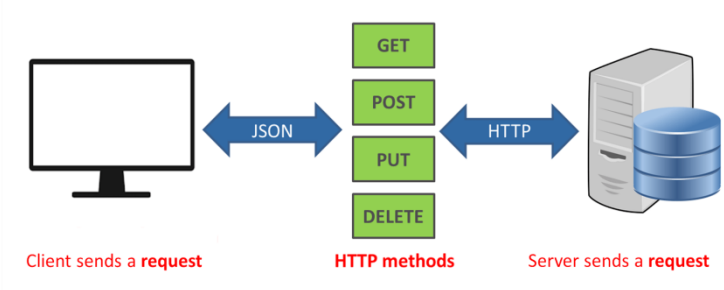

Рис.2.2 - графічне зображення роботи RESTful додатку

2.2.1 Spring Framework

У ході проектування дипломної роботи було обрано фреймворк Spring як основну платформу для розробки додатку. Пропоную розглянути його більш детально.

Spring Framework являє собою своєрідний open-source фреймворк, який має контейнери з підтримкою IoC (інверсії управління) для платформи Java.

Основні функції даного фреймворка можуть бути використані будь-якою програмою написаною на мові Java, але для створення веб-додатків Java EE є своє розширення. Не дивлячись на це, Spring не нав'язує якоїсь конкретної моделі для програмування, він став дуже популярним в спільноті Java як альтернатива, або навіть доповнення моделі EJB(Enterprise JavaBean).

Фреймворк забезпечує вирішення завдань, з якими стикаються Java розробники і бізнес, який хоче створити інформаційну систему, засновану на платформі Java. Через широку функціональність буває дуже важко визначити найбільш значущі структурні елементи, з яких він складається. Spring не цілком пов'язаний з платформою Java EE, незважаючи на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring Framework пропонує розробнику послідовну модель і робить її придатною для більшості типів додатків, які вже створені на основі платформи Java. Вважається, що даний фреймворк реалізує модель розробки, засновану на кращих стандартах індустрії, і робить її доступною в багатьох сферах мови Java.

Його основою є неймовірно спрощений контейнер елементів, який реалізує шаблон програмування IoC (Inversion of Control). Використовуючи цей контейнер, елементи не будуть створювати свої екземпляри і не будуть шукати свої залежності. Контейнер елементів відповідає за введення тих залежностей при створенні самих елементів — звідси і термін "Інєкція Залежностей" (Dependency Injection), який може також використовуватися для опису цього шаблону програмування.

Spring IoC контейнер виявився міцним фундаментом для створення стійких корпоративних додатків, який полегшує взаємодію між його елементами.

Елементи, які керуються Spring IoC контейнером називають бінами. Також, Spring framework містить в собі декілька інших модулів, наприклад: підтримку управліннями транзакцій, доступ до даних JDBC та ORM.

2.2.2 Модуль Spring MVC

Для реалізації трирівневої архітектури проекту я обрав модуль фреймворка Spring - Spring MVC.

Фреймворк Spring MVC забезпечує архітектуру патерна Model — View — Controller (Модель — Відображення (далі — вид) — Контролер) за допомогою слабко зв'язаних готових компонентів. Патерн MVC розділяє аспекти додатку (логіку вводу, бізнес-логіка та логіка UI), забезпечуючи при цьому вільний зв’язок між ними.

- Model (Модель) інкапсулює (об’єднує) дані додатку, в цілому вони будуть складатися з POJO.
- View (Відображення, Вид) відповідає за відображення даних моделі — як правило, генеруючи HTML, який ми бачимо в своєму браузері.
- Controller (Контролер) опрацьовує запит користувача, створює відповідну Модель та передає її для відображення у Вид.

Уся логіка роботи Spring MVC побудована довкола DispatcherServlet, який приймає і опрацьовує усі HTTP-запити (із UI) та відповіді на них. Робочий процес опрацювання запиту DispatcherServlet'ом проілюстрований на наступній діаграмі:

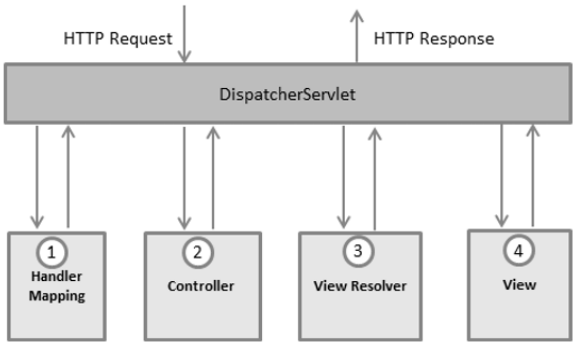

Рис.2.3 - графічне зображення роботи модуля Spring MVC

Нижче наведена послідовність подій, відповідна до вхідного HTTP-запиту:

- Після отримання HTTP-запиту DispatcherServlet звертається до інтерфейсу HandlerMapping, який визначає, який Контролер повинен бути визваний, після чого відправляє запит в необхідний Контролер.
- Контролер приймає запит та викликає відповідний службовий метод, оснований на GET чи POST. Викликаний метод визначає дані Моделі, заснованої на певній бізнес-логіці та повертає в DispatcherServlet ім’я Вида (View).
- За допомогою інтерфейсу ViewResolver DispatcherServlet визначає, який Вид потрібно застосовувати на основі отриманого ім’я.
- Після того, як Вид (View) створений, DispatcherServlet відправляє дані Моделі у якості атрибутів у Вид, який у кінцевому результаті відображається у браузері.

Усі вищезгадані компоненти, а саме HandlerMapping, Controller и ViewResolver, є частинами інтерфейсу WebApplicationContext extends ApplicationContext з деякими додатковими особливостями для створення web-додатків.

2.2.3 Модуль Spring Data

Для взаємодії з сутностями БД, організації їх у репозиторії, витягу даних, редагування я обрав модуль Spring Data. Велика перевага даного модуля в тому, що в багатьох випадках для усіх вищезазначених операцій буде достатньо лише оголосити інтерфейс і метод у ньому, без імплементації.

Основне поняття у Spring Data - це репозиторій. Це декілька інтерфейсів, які використовують JPA Entity для взаємодії з ним. Так, наприклад інтерфейс

```
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID>
```

надає основні операції по пошуку, зберіганню, видаленню даних (CRUD операції)

```
T save(T entity);
Optional findById(ID primaryKey);
void delete(T entity);
```

ці, та інші.

Тобто, якщо того переліку, що надає інтерфейс достатньо для взаємодії з сутністю, то можно розширити базовий інтерфейс для своєї сутності, доповнити його своїми методами запитів та виконувати певні операції.

Запити до сутності можна будувати прямо з імені метода. Для цього використовується механізми префіксів find…By, read…By, query…By, count…By, та get…By. Далі від префікса метода починається розбір решти. Вступне речення може містити додаткові вираження. Наприклад, Distinct. Далі, перший By працює як роздільник, щоб вказувати початок фактичних критеріїв. Можна визначати умови для властивостей сутностей та об’єднувати їх за допомгою And та Or. Наприклад:

```
@Repository
public interface CustomizedEmployeesCrudRepository extends CrudRepository<Employees, Long> {
    // шукати по полям firstName And LastName
    Optional<Employees> findByFirstNameAndLastName(String firstName, String lastName);
```

```
    // знайти перші 5 по FirstName, які починаються з символів та сортувати по FirstName
    List<Employees> findFirst5ByFirstNameStartsWithOrderByFirstName(String firstNameStartsWith);
```

В документації визначений весь перелік та правила написання методів. У якості результата може бути сутність T, Optional, List, Stream. У середовищі розробки IDEA, є підказки щодо написання даних методів.

2.2.4 Модуль Spring Security

Для аутентифікації та авторизації користувачів був обраний модуль Spring Security, який є одним з найбільш простих та водночас ефективних інструментів.

У порівнянні з основним конкурентом – Apache Shiro, Spring Security виграє в гнучкості, простоті написання та модульності. Тож, розглянемо даний модуль більш детально.

Spring Security - це Java/JavaEE фреймворк, який надає механізми побудови систем аутентифікації та авторизації, а також інші можливості забезпечення безпеки для корпоративних додатків, створених за допомогою Spring Framework.

Розглянемо ключові об’єкти контекста Spring Security.

Перш за все, варто згадати об’єкт SecurityContextHolder, адже в ньому міститься інформація по поточному контексті безпеки додатку, який включає в себе детальну інформацію про користувача (Principal), який працює в реальному часі з додатком. За замовчуванням SecurityContextHolder використовує ThreadLocal для зберігання такої інформації. Що означає, що контекст безпеки завжди доступний для методів, які виконуються в тому ж самому потоці.

Контекст SecurityContext, містить в собі об’єкт Authentication і у разі необхідності інформації системи безпеки, яка зв’язана з запитом від користувача.

Інтерфейс Authentication представляє користувача (Principal) з точки зору Spring Security, а GrantedAuthority відображає дозволи, видані користувачу у масштабі усього додатку. Такі дозволи, як ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN.

Після ознайомлення з основними контекстами Spring Security варто звернути увагу на його алгоритм аутентифікації:

1. Користувачу буде запропоновано увійти до систему надавши ім’я (логін або email) та пароль. Ім’я користувача та пароль об’єднуються у копію класа UsernamePasswordAuthentication (копія інтерфейсу Authentication) після чого він передається копії AuthenticationManager для перевірки.
2. У разі, якщо пароль не відповідає оригіналу, буде викинуто виключення BadCredentialsException з повідомленням “Bad Credentials”.
3. Якщо аутентифікація пройшла успішно, повертається повністю заповнена копія Authentication.
4. Для користувача встановлюється відповідний контекст безпеки шляхом виклику метода SecurityContextHolder. setAuthentication(), куди передається об’єкт, який повернув AuthenticationManager.

2.2.5 Мова програмування Java

Для написання серверної частини проекту мною було обрано мову програмування Java.

Java — це ООП мова програмування, яка була створена у 1995 році компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія Oracle. Компанія надає компілятор Java та віртуальну машину Java, які задовольняють специфікації JCP (Java Community Process), під ліцензією GNU (General Public License).

Після визначення основних архітектурних концепцій для шару додатку, переді мною постало питання вибору інструменту для імплементації продукту. Було обрано середовище для розробки програмного забезпечення IntelliJ IDEA.

Даний інструмент надає зручні засоби для розробки додатків на Java, можливість створювати консольні, десктопні, мобільні та веб-додатки. Основною перевагою перед своїми конкурентами, є його доступність не тільки на Windows, а й на Linux та MacOS.

Інструмент надає широкі можливості для навігації, виправлення та переробки коду. Однією з головних переваг, є наявність вбудованого терміналу та системи контролю версій. Також у даний продукт вбудовано профайлер для відслідковування проблем з продуктивністю розробленого додатку.

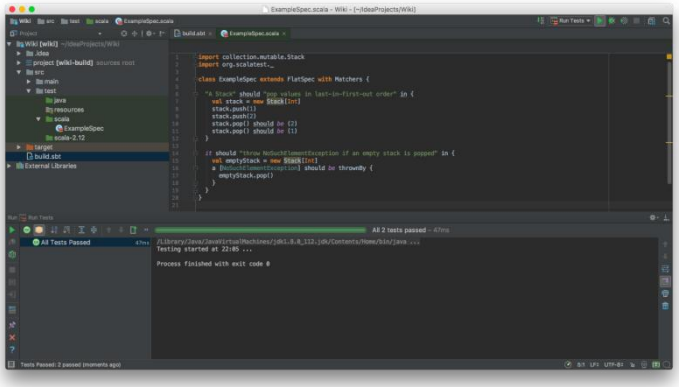

Рис. 2.4 – Зовнішній вигляд IDE Intellij IDEA

2.3 Проектування моделі бази даних

Для зручності та ефективної роботи з даними в першу чергу, потрібно визначити об'єктну модель системи та бази даних. Ця модель використовується для опису даних, які використовуватимуться системою.

В першому розділі надавалася деяка інформація щодо моделі даних DDF, яка була обрана для правильної організації БД. Розглянемо її більш детально.

2.3.1 Датапоінти в моделі даних DDF

Дані у DDF зберігаються у форматі пар “ключ-значення” які називаються Датапоінти. Ключ складається з двух або більше аспектів, тоді коли значення має лише 1 індикатор. Нижче наведена таблиця, яка показує:

1. Датапоінти з ключем: «країна» та «рік» та значенням: «тип правління»
2. Датапоінти з ключем: «країна» та «рік» та значенням: «населення»

Таблиця 2.1 – Табличне зображення датасету

	Ключ: Аспекти		*Значення: Індикатори*	
Концепти	**країна**	**рік**	**Тип_правління**	**населення**
Датапоінти	Німеччина	1940	Диктатура	3954858
	Німеччина	2015	Парламентська демократія	50495854
	Швеція	2015	Парламентська демократія	9504847

В DDF Датапоінтах:

- Аспект - це є концепт “ключа” у парі “ключ-значення” Датапоінта. У статистиці, концепт “ключа” зветься незалежною змінною.

  У таблиці country and year - це аспекти, які формують “ключ” для індикаторів.
- Індикатор - це концепт “значення” у парі “ключ-значення” Датапоінта. У статистиці, індикатор зветься залежною змінною, бо його значення залежить від “ключа”. У таблиці «тип_правління» та «населення» індикатори.

Концепт може бути як аспектом, так і індикатором. У таблиці нижче наведено приклад як концепт «лідер» виступає і аспектом, і індикатором.

Таблиця 2.2а – Табличне зображення концепту як індикатора

	Ключ:Аспекти		*Значення: Індикатори*
Концепти	**країна**	**рік**	**лідер**
Датапоінти	Germany	2015	Angela Merkel

Таблиця 2.2б – Табличне зображення концепту як аспекта

	Ключ:Аспекти		*Значення: Індикатори*
Концепти	**лідер**	**рік**	**виборчий рейтинг**
Датапоінти	Ангела Меркель	2015	0.48

2.3.2 Аспекти та індикатори в моделі даних DDF

Аспекти формують “ключ” у Датапоінті. Датапоінти в DDF часто використовуються для спостереження за змінами протягом багатьох років та

місць на карті. Можна додати будь-яку кількість аспектів, щоб краще уточнити ключ. Наприклад:

Таблиця 2.3 – Табличне зображення великої кількості аспектів

Ключ:Аспекти				*Значення: Індикатори*
країна	**рік**	**стать**	**вікова категорія**	**населення**
Швеція	2015	чол	0-4	485 055
Швеція	2015	жін	0-4	483 538

Варто зазначити, що DDF не строгий до часу то локації. Будь-який індикатор можна зберігати у будь-якому наборі аспектів.

Набір індикаторів можна розширити, щоб надати більше даних. У таблиці, наведеній нижче, дані про населення та тривалість життя надані декільком аспектам: країні та рокам.

Таблиця 2.4 – Табличне зображення розширення кількості індикаторів

Ключ:Аспекти		*Значення: Індикатори*	
країна	**рік**	**населення**	**середня тривалість життя**
Швеція	2015	9 838 480	81.48
Швеція	2014	9 644 864	81.38

Індикатори не повинні бути цілим числом, або взагалі навіть числом. В таблиці нижче показано концепт «тип_правління», в якому індикатор є словом та належить до певної категорії.

Таблиця 2.5 – Табличне представлення індикатора у форматі String

Ключ:Аспекти		*Значення: Індикатори*
країна	**рік**	**тип_правління**
Німеччина	1940	диктатура
Німеччина	1990	парламентська демократія

Оскільки, DDF не строгий до часу та геолокації, то в ньому можна зберігати будь-який тип індикаторів:

Таблиця 2.6 – Табличне зображення індикаторів різних форматів

Ключ:Аспекти		*Значення: Індикатори*
мова_програмування	**компанія**	**кількість_проектів**
C#	microsoft	405
C#	apple	949
Javascript	microsoft	6594

2.3.3. Концепти у моделі даних DDF

Концепти є заголовками таблиць у будь-якому місці набору даних DDF. Властивості концептів - це також концепти, які дають інформацію про них самих.

Тип концепту визначає тип даних у значеннях під цим концептом.

Типи, які можно застосовувати в DDF: *string, measure* (numeric), *boolean, interval, time* та більш DDF-орієнтовані *entity domain, entity set ma role*.

Будь-яка властивість будь-якого концепту може бути визначено наступним чином в DDF. Усі концепти, які використовуються у датасеті повинні бути пронумеровані, а усі властивості цього концепту повинні бути визначеними. Також, *ідентифікатори концептів можуть містити лише літери у нижньому регістрі, цифри та нижнє підкреслення.*

2.3.4. Метадані у моделі даних DDF

Метадані - це інформація про дані. Наприклад: чоловіче населення Швеції у 2015 році - 99483857. Це - дані. Метаданими для цих даних будуть її джерело (статистика населення ООН), точність (5%) та певні примітки. Жіноче населення Швеції в 2015 році може мати інше джерело, точність та примітку. Те ж саме для цих самих даних за 2016 рік.

Приклад, наведений нижче, демонструє як метадані можуть описувати одне певне число у датасеті, одне певне значення у датапоінті. Тому, в DDF ми визначаємо метадані як властивість однієї пари ключ-значення даних у колекції. У таблиці нижче всі 4 комірок у стовпцях “значення” можуть мати свої власні, абсолютно різні метадані.

Таблиця 2.10 – Табличне зображення метаданих

key			*value*	*value*
country	**year**	**gender**	**population**	**income_p_cap**

<table>
<tr>
<td>Sweden</td>
<td>2015</td>
<td>male</td>
<td>9 9483 857
<table>
<tr><th>note</th><th>acc</th><th>src</th></tr>
<tr><td>foo</td><td>95%</td><td>UN</td></tr>
</table>
</td>
<td>48958749
<table>
<tr><th>note</th><th>acc</th></tr>
<tr><td>bar</td><td>92%</td></tr>
</table>
</td>
</tr>
<tr>
<td>Sweden</td>
<td>2015</td>
<td>female</td>
<td>483 538
<table>
<tr><th>note</th><th>src</th></tr>
<tr><td>bar</td><td>UN</td></tr>
</table>
</td>
<td>49898383
<table>
<tr><th>note</th><th>acc</th><th>src</th></tr>
<tr><td>bar</td><td>92%</td><td>WB</td></tr>
</table>
</td>
</tr>
</table>

Однак, можливо, що поля містять однакові метадані. Наприклад, у наведеній вище таблиці всі дані про населення мають одне й те саме джерело.

2.3.5. Обгрунтування вибору БД та інструментів

Після розгляду та проектування конкретної моделі даних для проекту, потрібно було обрати базу даних, яка б задовільняла основні потреби додатку.

Після аналізу наявних рішень було вирішено обрати як основну – реляційну БД, адже вони працюють із структурованими даними та підтримують ACID транзакції, добре підтримують зв’язки та їх обмеження, механізми індексації, використовують SQL, як основну мову для написання запитів, Недолік у вигляді поганого горизонтального розширення не є суттєвим в даному випадку.

У підсумку, з усіх реляційних БД я обрав MS SQL Server. В неї є можливість створення, зміни та управління даними. У MS SQL Server

присутні функції для створення звітів, імпортування та експорту даних, інструменти для аналізу даних.

Для підтримки функціоналу, який надається SQL Server було створено спеціальне розширення – T-SQL. Воно надає можливість процедурного програмування та створення локальних змінних, розширених засобів для роботи з рядками, датами та математичними виразами. Також серед важливих змін треба обов’язково відзначити додавання операторів для роботи з BULK-операціями, можливість обробки даних в стилі ООП мов програмування таких як Java та C#, із застосування ключових слів TRY..CATCH.

Також, перевагою SQL Server є те, що він кросс-платформовий та може бути розгорнутий не лише на Windows, а й Linux. Також SQL Server уже використовують в контейнеризованих системах.

Висновки до розділу

1. Було обрано архітектуру та платформу на якій буде здійснюватися розробка.
2. Опис архітектури та допоміжних засобів, які потрібні для розробки системи.
3. Спроектовано серверну та клієнтську частини.
4. Спроектовано модель бази даних для нашої системи.

Отже, для розробки було обрано трирівневу архітектуру. Обрано саме її через розділення програми на три рівні, що дає змогу легко масштабувати систему та легко вносити до неї зміни. Для реалізації цього, також, обрано платформу Spring Framework та мову програмування Java. Для роботи з базою даних вибрано Spring Data, він дає можливість легко взаємодіяти з БД.

Для оформлення представлення (View) обрано фреймворк ReactJS та мову програмування JavaScript.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СПРОЕКТОВАНИХ ПІДСИСТЕМ, ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація спроектованих підсистем

В попередньому розділі було спроектовано систему та її підсистеми, тож тепер можемо приступити до її розробки.

3.1.1 Реалізація моделі бази даних

При описі засобів побудови системи було вказано, що в системі використовується спеціальна ORM бібліотека Hibernate. При використанні

даної бібліотеки я використав метод створення моделі за готовою базою даних, яку було заздалегідь створено.

Опис таблиць, які входять в у БД представлено на табл. 3.1-3.7:

Таблиця 3.1 – Користувач (SiteUsers)

Назва	Тип	Опис
Site_User_ID	int	Унікальний ідентифікатор користувача
User_Name	nvarchar	Логін
Password	nvarchar	Пароль
Site_User_Role_ID	int	Ідентифікатор ролі користувача
Email	nvarchar	Адреса електронної скриньки

Таблиця користувачів, окрім основних даних містить зовнішній ключ Site_User_Role_ID для зв'язку з таблицями ролей. Також, варто зазначити, що усі поля данної таблиці не можуть бути null.

Таблиця 3.2 – Роль користувача (SiteUserRoles)

Назва	Тип	Опис
Site_User_Role_ID	int	Унікальний ідентифікатор ролі користувача
Name	nvarchar	Назва ролі
Description	nvarchar	Назва ролі українською

Дана таблиця містить дані про ролі користувачів в системі. Їх в системі усього три: ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN. Варто зазначити, що користувачеві з роллю ROLE_ANONYMOUS буде доступна лише сторінка з авторизацією.

По опису моделі даних DDF в попередньому розділі, спроектуємо архітектуру бази даних для збереження самих статистичних даних.

Опишемо таблицю Concepts.

Таблиця 3.3 – Концепти (Concepts)

Назва	Тип	Опис
Concept_ID	int	Унікальний ідентифікатор концепту
Concept_Name	nvarchar	Назва концепту

Дана таблиця зберігає в собі назви концептів (фактично заголовків таблиц, їх назви) і не має зовнішніх ключів на інші таблиці.

Опишемо таблицю Aspects.

Таблиця 3.4 – Аспекти (Aspects)

Назва	Тип	Опис
Aspect_ID	int	Унікальний ідентифікатор аспекту
Concept_ID	int	Ідентифікатор аспекту
Data_Point_ID	int	Ідентифікатор датапоінту
Meta_Data_ID	int	Ідентифікатор метаданих
Value	nvarchar	Значення аспекту

Таблиця аспектів описує значення концептів з попередньої таблиці та формує список усіх аспектів. Дана таблиця має зовніші ключі на таблиці Concepts, DataPoints, Meta_Data.

Опишемо таблицю DataPoints.

Таблиця 3.5 – Датапоінти (DataPoints)

Назва	Тип	Опис
Data_Point_ID	int	Унікальний ідентифікатор датапоінту
Data_ID	int	Ідентифікатор таблиці статистичних даних

За допомогою таблиці DataPoints формується список аспектів. Таблиця має зовнішній ключ на таблицю Data, що зберігає набір дата-поінтів і формує статистичний набір даних.

Опишемо таблицю Data.

Таблиця 3.6 – Статистичні дані (Data)

Назва	Тип	Опис
Data_ID	int	Унікальний ідентифікатор статистичного набору даних
Name	nvarchar	Назва набору даних
Creator_ID	id	Ідентифікатор користувача, що створив даний набір даних

Таблиця Data потрібна для формування набору датапоінтів, а також для формування статистичного набору даних. Таблиця має зовнішній ключ на таблицю Users.

Опишемо таблицю MetaData.

Таблиця 3.7 – Метадані (MetaData)

Назва	Тип	Опис
Meta_Data_ID	int	Унікальний ідентифікатор мета-даних
Note	nvarchar	Опис
Acc	id	Точність статистичних даних
Src	nvarchar	Джерело статистичних даних

Таблиця MetaData описує метадані статистичних даних, фактично – містить інформацію про статистичні дані для користувача. За замовченням поле Acc та Src не можуть бути null(порожніми), а поле Note – може. Воно містить деяку додаткову інформацію, яку користувач може заповнювати за бажанням при додаванні до системи.

Опишемо таблицю Sinonimus і Words. Ці таблиці формують масиви синонімів, що спрощують пошук.

Таблиця 3.8 – Синоніми (Sinonimus)

Назва	Тип	Опис
Sinonimus_ID	int	Унікальний ідентифікатор метаданих

Таблиця 3.9 – Слова (Words)

Назва	Тип	Опис
Meta_Data_ID	int	Унікальний ідентифікатор метаданих
Note	nvarchar	Опис
Acc	id	Точність статистичних даних
Src	nvarchar	Джерело статистичних даних

По описаним вище таблицям можна зформувати структурну діаграму бази даних.

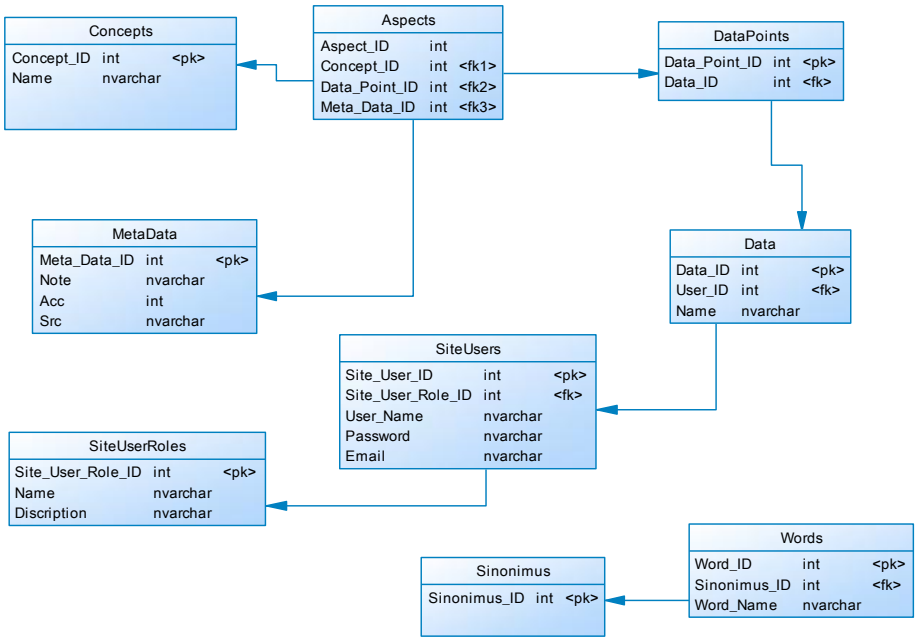

Рисунок 3.1 – Діаграма структурна бази даних

3.1.2 Реалізація серверної частини системи

Як було зазначено вище у попередніх розділах, реалізація серверної частини базується на архітектурі шаблону MVC, тому система складається з трьох частин: моделей, представлень та контролерів.

Під час роботи користувача з системою, веб-браузер посилає запити на сервер, які далі передаються на обробку. За обробку цих запитів відповідають контролери, а саме, окремі методи в цих контролерах.

Розглянемо діаграму класів реалізованого веб-сервісу.

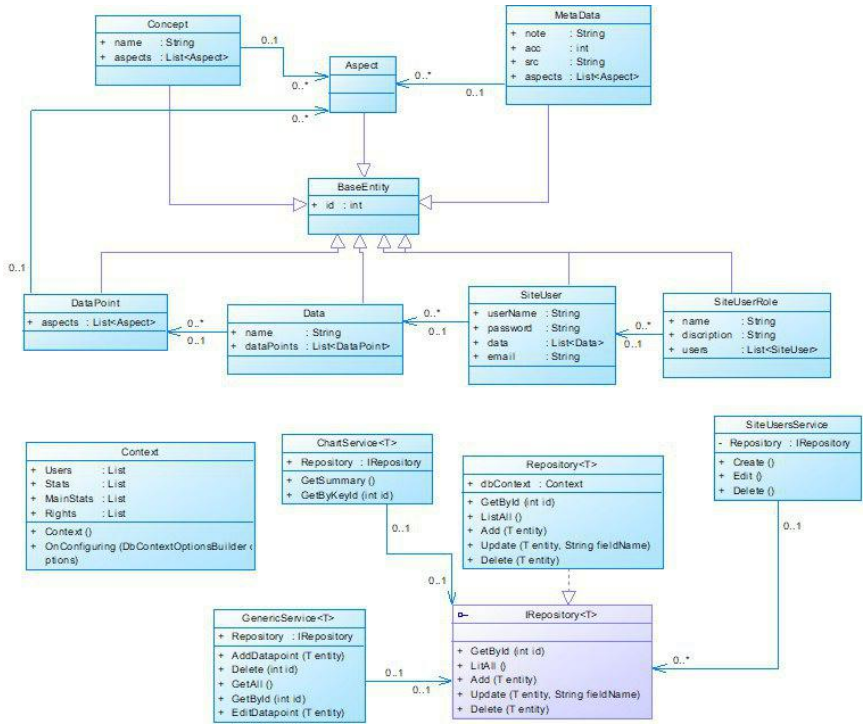

Рисунок 3.2 – Діаграма структурна бази даних

Зокрема, якщо взяти для прикладу роботу з користувачами, то для цього є спеціальний клас «SiteUsersController», вихідний код якого представлений у додатку А. Даний контролер та всі інші, наслідуються від контролера BaseController, який наслідується від класу бібліотеки MVC «Controller».

Клас «SiteUsersController» містить 9 методів. Дані методи відповідають за обробку запитів та формування правильної відповіді сервера, також вони описують специфічну функціональність для кожної з сторінок представлення. Для правильної функціональності роботи з користувачами є такі дії:

- відображення головної сторінки, яка містить інформацію про всіх користувачів (Index);
- відображення сторінки перегляду деталей користувача (Details);
- створення нового користувача (Create);

- редагування вибраного користувача (Edit);
- видалення обраного користувача (Delete).

В контролері для створення, редагування та видалення створено по 2 дії, з однаковою назвою та різним функціоналом. Це зроблено для того, щоб уникнути атак підміни запитів, та називається перевантаженням методів. Найчастіше вони відрізняються параметрами, які будуть надходити до цих методів. Наприклад, в контролері «SiteUsersController» існує дві дії для редагування користувача («Edit»). Перша дія в якості параметра приймає Id користувача для редагування того користувача, який було обрано та оброблює запит Get. Друга дія в якості параметра приймає клас моделі «SiteUser», тобто екземпляр класу з уже заповненими полями, який було отримано з представлення, також, ця дія являється запитом Post. Get запити відрізняються від Post запитів тим, що параметри в запитах Post передаються не в адресному рядку, а в самому тілі запиту, тому дані запити являються більш безпечними та використовуються для передачі багатьох параметрів.

Для роботи з даними було створено клас DataPointsService, який реалізовує основні CRUD (create-read-update-delete) операції з даними. Дані методи моужть використовувати контролери, загалом, контролер DatapointsController, що приймає запити з клієнтської частини на читання, редагування, видалення, оновлення.

За зчитування даних відповідають методи GetAll() та GetById(int id). Перший повертає увесь масив даних, а другий повертає дані по вказаному id.

Коли клієнт посилає запит на оновлення даних з об’єктом, система перевіряє коректність даних. Після чого викликається метод GetAll() і перевіряється наявність цих даних. Якщо дані відсутні, то відбувається додавання даних. За додавання даних відповідає метод AddDatapoint(T entity). Метод приймає об’єкт, мапить його та створює запис в базі даних. Якщо ж дані наявні, то відбувається редагування даних. За редагування

даних відповідає метод EditDatapoint (T entity). Метод приймає сам об’єкт, на який буде замінено запис в БД.

За видалення даних відповідає метод Delete(int id). Метод приймає id запису, що потрібно видалити і видаляє його каскадно.

Візуалізація графіків і діаграм була реалізована за допомогою бібліотеки Recharts до фреймворку React. Ця бібліотека дозволяє візуалізовувати масиви інформаціїї різними способами. Для отримання масиву даних, клієнт посилає запит на сервер і у відповідь, сервер посилає відповідну json-відповідь. Клієнт парсить json і перетворює його в масив об'єктів. В кожному об'єкті вказано назву ключа (наприклад 2012 рік.) і відповідне значення до цього ключа (зазвичай число). Цей масив передається в компонент Rechart після чого графік візуалізовується.

Для роботи з графіками було реалізовано сервіс ChartService, який реалізовує методи, що повертають масиви даних, для візуалізації графіків та діаграм. В сервісі описані таки методи як GetSummary, GetByKeyId.

GetSummary повертає підсумовані дані по всім ключам.

GetByKeyId повертає дані по відповідно заданому ключу.

3.1.3 Реалізація клієнтської частини системи

При відкритті веб-застосування користувач в залежності від того, авторизований він чи ні відкривається відповідна сторінка.

Для не авторизованого користувача доступна сторінка авторизації та реєстрації. Якшо користувач не зраеєстрований, він зможе зареєструватись, а потім авторизуватись.

Для авторизованого користувача з роллю «Системний адміністратор» доступні такі сторінки:

1. Управління статистичними даними - додавання, редагування, видалення та детальний перегляд статистичних даних.
2. Валідація статистичних даних, завантажених користувачами - завантаження, видалення та підтвердження проходження валідації даних.
3. Управління користувачами системи - видалення та перегляд інформації про користувача.

Для авторизованого користувача з роллю «Користувач» доступні такі сторінки:

1. Головна сторінка - пошук необхідних статистичних даних за ключовими словами або категоріями, відображення історії переглядів користувача, додавання власних статистичних даних у систему.
2. Сторінка перегляду статистичної інформації - перегляд статистичних даних візуалізованих за допомогою графіків, завантаження статистики у табличному форматі, використання інструменту для певної вибірки статистичних даних.

Розглянемо детальніше:

1. Сторінка «Реєстрація»

Сторінка реєстрації (Рис. 3.1) потрібна для створення нового облікового запису користувача та подальшого входу до системи за допомогою попередньо введених даних. Після заповнення потрібних полів відбувається виконання запиту та виклик дії «Create» контролеру «AccountsController». Далі ці дані перевіряються на відповідності правилам валідації, якщо все гаразд – відбувається збереження даних користувача в базі даних через контекст. Після цього, користувач вважається створеним.

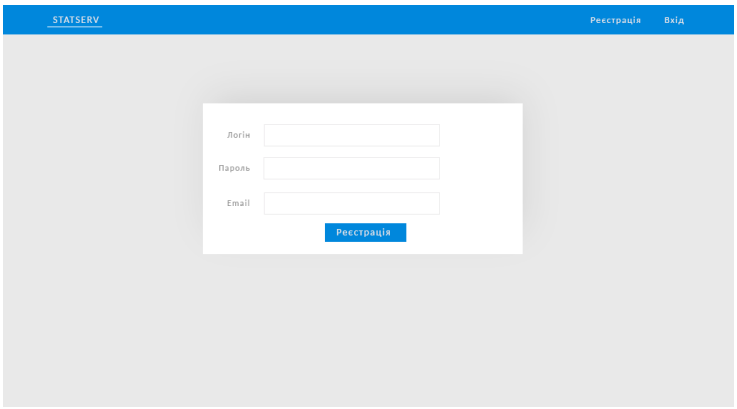

Рис. 3.1 – Сторінка реєстрації нового користувача

2. Сторінка «Вхід»

Для зареєстрованого та неавторизованого користувача доступна сторінка «Вхід» (Рис. 3.2). Після вводу правильних даних в потрібні поля, програма надає доступ до системи. Для різних типів користувачів доступні сторінки, які відповідають правам користувачів.

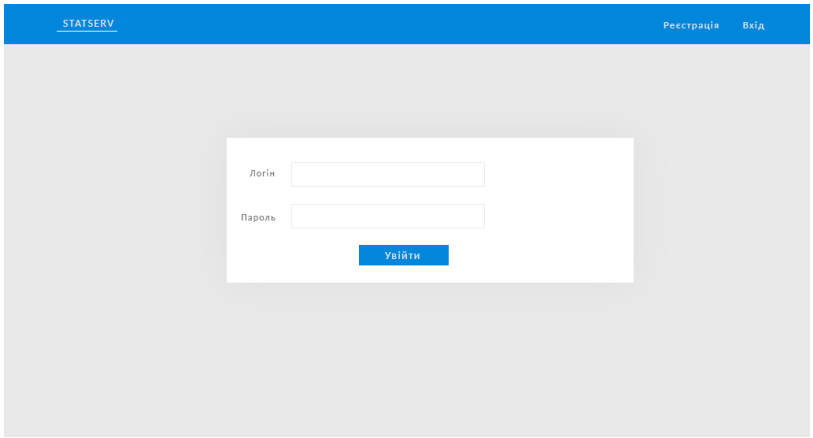

Рис. 3.2 – Сторінка авторизації користувачів

3. Сторінка «Головна»

Після вдалого входу до системи з роллю «Користувач», користувач потрапляє на головну сторінку. Тут він може здійснити пошук за ключовими словами, категоріями, продивитися свою історію переглядів чи завантажити свої статистичні дані.

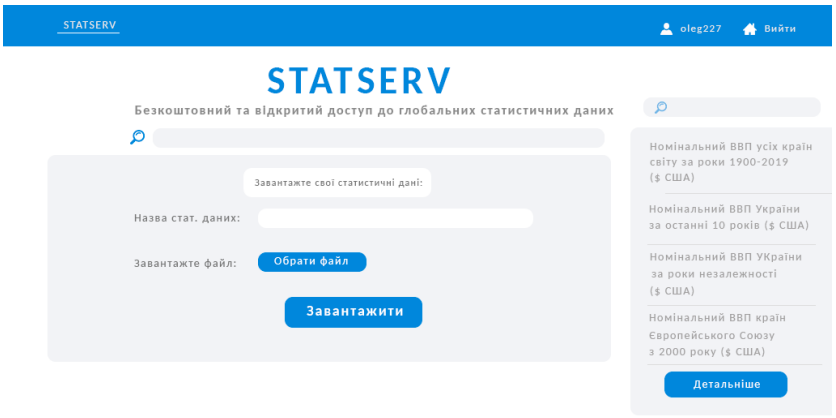

Рис. 3.3 – Головна сторінка сервісу

4. Сторінка «Перегляд статистики»

Також для користувача з роллю «Користувач» доступна сторінка перегляду статистичних даних. Тут він може побачити статистику у графічному вигляді, завантажити статистичні дані та зробити вибірку даних.

Рис. 3.4 – Сторінка перегляду статистичних даних

5. Сторінка «Користувачі»

На даній сторінці у користувачів з роллю «Системний адміністратор» є можливість переглянути інформацію, редагувати та видаляти користувачів з системи, які є в БД (Рис. 3.5).

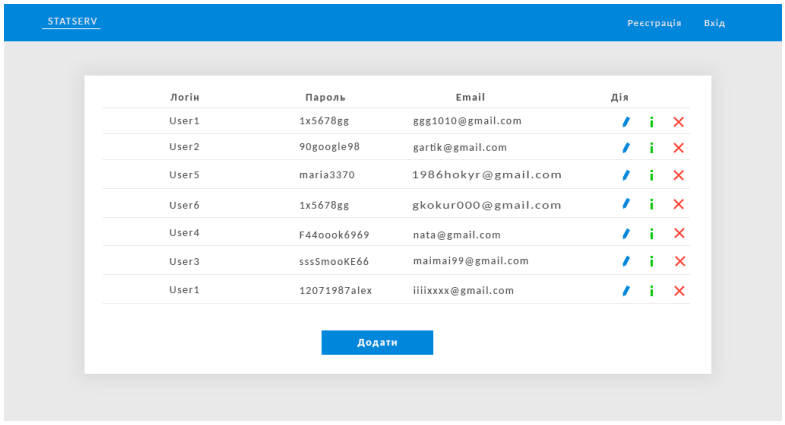

Рис. 3.5 – Сторінка перегляду зареєстрованих користувачів

6. Сторінка «Управління статистичними даними»

На даній сторінці у користувачів з роллю «Системний адміністратор» є можливість переглянути інформацію, редагувати та видаляти статистичні дані з системи, які є в БД (Рис. 3.6).

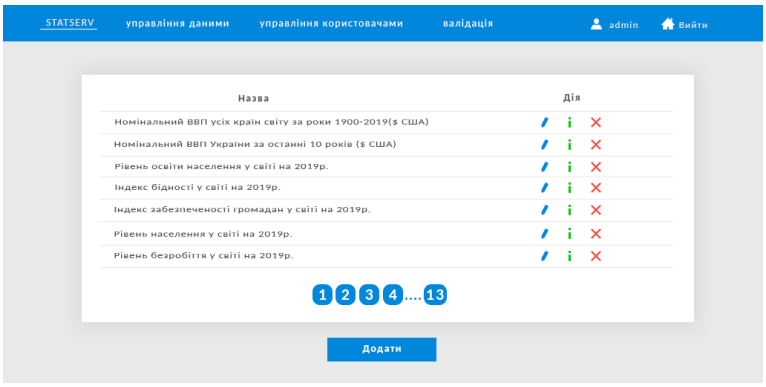

Рис. 3.6 – Сторінка управління статистичними даними

7. Сторінка «Валідація завантажених даних користувачами»

На даній сторінці у користувачів з роллю «Системний адміністратор» є можливість переглянути інформацію, редагувати та видаляти статистичні дані з системи, які є в БД (рис. 3.7).

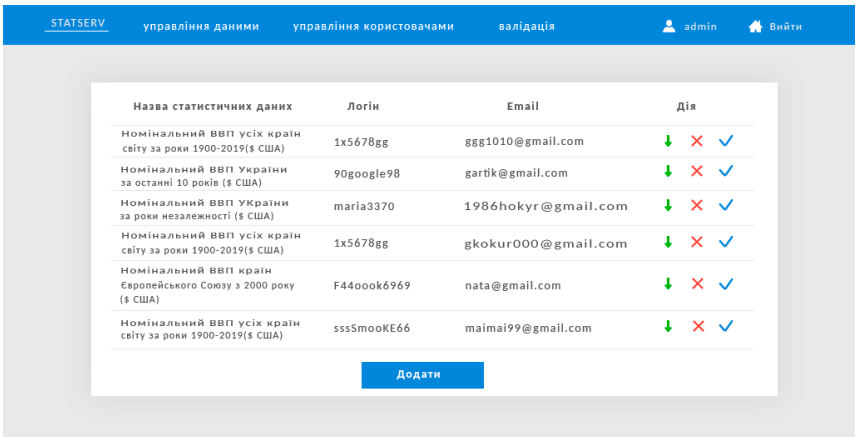

Рис. 3.7 – Сторінка валідації завантажених даних користувачами

3.2. Тестування системи

Для виявлення помилок в готовій програмі відбувається тестування сервісу. Існує багато видів тестування програмного забезпечення, та для тестування даної системи було обрано ручне та модульне тестування. Про них і піде мова далі.

3.2.1 Ручне тестування

Ручне тестування представляє собою процес пошуку різних помилок під час роботи програмного забезпечення. Під час пошуку помилок моделюються різні сценарії дій користувача. В нашому випадку тестувальник перевіряє працездатність програми імітуючи дії звичайного користувача. Для більш точної перевірки використовують завчасно підготовлений план тестування, де перераховані найбільш важливі частини застосування.

В даному випадку, наведений нижче план тестування дозволить перевірити деякі "Слабкі місця" нашої програми.

Тестування правильності введених значень.

а) тестування на відсутність даних;

б) тестування на введення некоректних даних;

в) тестування на введення неіснуючих даних.

В проекті будемо перевіряти наведені вище кроки на наступних сторінках:

а) сторінка входу;

б) сторінка реєстрації.

1) Виконаємо перевірку для сторінки входу в систему.

На даній сторінці присутні 2 поля: логін та пароль користувача. При спробі входу без необхідних даних вхід не повинен виконуватися та повинні показуватися відповідні повідомлення. Результат виводу при спробі увійти не вказавши дані представлений на рис. 3.8

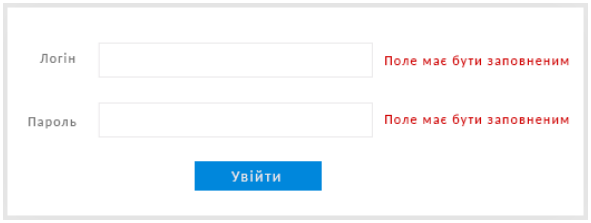

Рис. 3.8 – Результат виконання при відсутності даних в полях на сторінці входу

Також на сторінці присутнє поле введення паролю, яке має відповідний формат. Перевірка обробки неправильного введення паролю на даній сторінці представлена на рис. 3.9.

Рис. 3.9 – Результат виконання при неправильному введенні паролю на сторінці входу

На наступному етапі перевіримо реакцію застосування на спробу входу за допомогою неіснуючих в системі даних. Результат виконання представлений на рис. 3.10

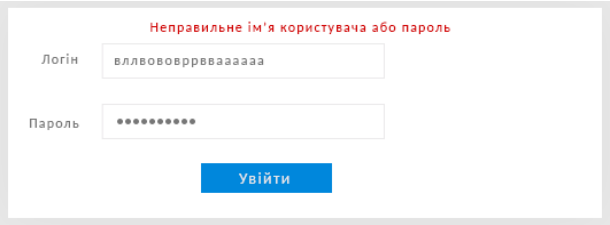

Рис. 3.10 – Результат виконання при введенні не існуючих в системі даних на сторінці входу

Як ми пересвідчились, сторінка входу адекватно обробляє некоректно введені дані і виводить відповідні повідомлення.

2) Тепер перевіримо сторінку реєстрації. На даній сторінці присутні 3 поля: логін, пароль користувача, та email. Обов'язковість полів тут присутня так як і на сторінці входу, також немає сенсу виконувати перевірку на неіснуючі дані тому що дана сторінка створює нового користувача при правильних даних. Результат перевірки на обов'язковість полів представлений на рис. 3.11.

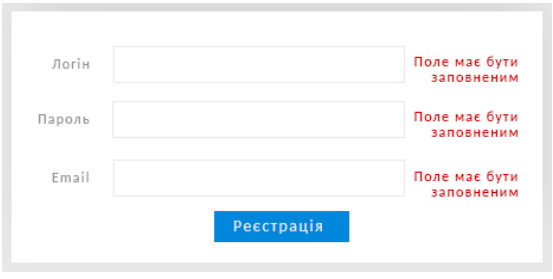

Рис. 3.11 – Результат виконання при відсутності даних в полях на сторінці реєстрації

Далі перевіримо поля «Логін», «Пароль», та «Email» на правильність вводу. Результат перевірки представлений на рис. 3.12.

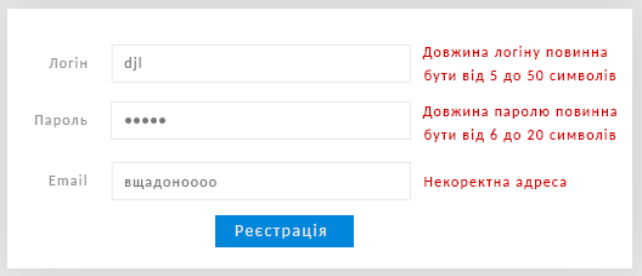

Рис. 3.12 – Результат виконання при неправильному введенні даних в полях «Логін», «Пароль», та «Email»

3.2.2 Модульне тестування

Модульні тести – це невеличкі програми, які дозволяють швидко протестувати окремі складові коду незалежно від інших частин застосування. Під час тестування вибираються частини коду, зазвичай це методи класів. Вони тестуються за допомогою відповідних засобів. Один з мінусів модульних тестів є те, що з часом їх стає надто багато. Тому процес проведення тестів автоматизують за допомогою відповідних фреймворків.

Виконаємо тестування для контролеру AccountsController в якому знаходяться методи методи «Login» та «Registration», саме ці методи і будемо тестувати. Вихідний код тесту зображений у додатку Б. Результат тестування зображено на рис. 3.13.

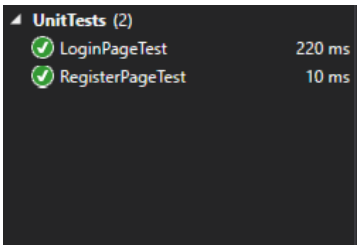

Рис. 3.13 – Результат виконання модульного тесту для контролеру AccountsController

Як ми можемо побачити на зображенні, тести виконуються успішно (відобразилися зелені позначки). У даному тесті виконувалася перевірка, чи згенерувалося потрібне представлення.

Типи тестів представляються трьома етапами відповідно до номеру рядка: початкова ініціалізація контексту тестів, виконання дії яку треба протестувати та перевірка, чи було створене потрібне представлення.

Додамо подібні тести для сторінки управління даними, результати тестування можна побачити на рис. 3.14.

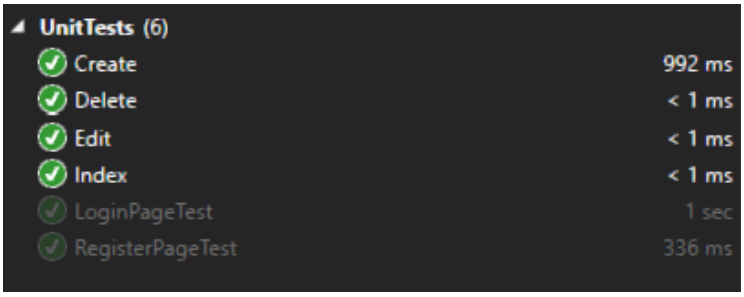

Рис. 3.14 – Результат виконання модульного тесту для управління статистичними даними

Як ми можемо пересвідчитись, модульне тестування є дуже корисним методом для перевірки реакції системи на значну кількість схожих вхідних даних, що дозволяє зекономити час розробника та провести якісне тестування.

3.3. Інструкція користувача

Перш ніж скористатися функціоналом системи потрібно зареєструватися в ній. Припустимо, що користувач вже зареєстрований. Тепер йому потрібно зайти в систему. Це можна зробити за допомогою форми, зображеної на рисунку 3.15.

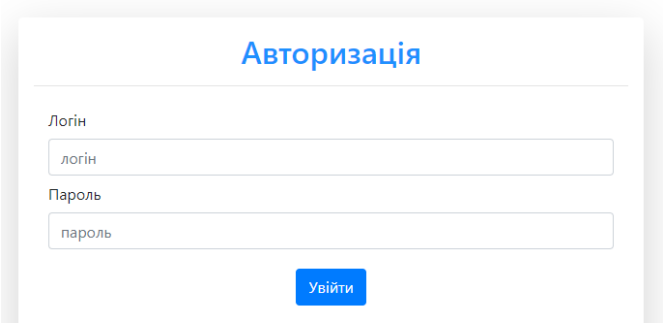

Рис. 3.15 – Форма входу в систему

Після вводу логіну та паролю і опинившись в системі, користувач буде спостерігати наступну сторінку (рис. 3.16), де буде зображено основний функціонал для пошуку необхідних статистичних даних.

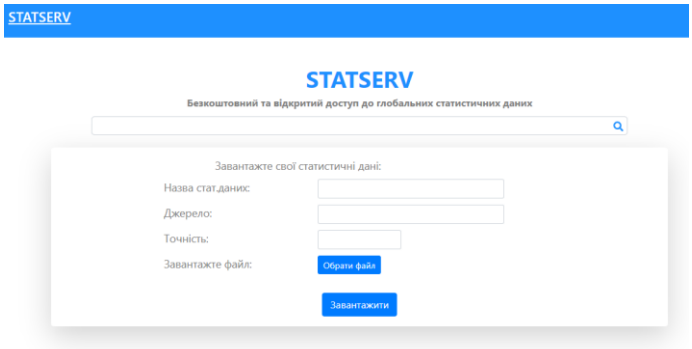

Рис. 3.16 – Головна сторінка сайту

Якщо користувач має бажання поповнити базу даних нашого сервісу йому потрібно заповнити усі поля та завантажити свій файл з даними у блоці нижче. Також, якщо користувач вже відвідував наш сервіс – він може подивитися свою історію переглядів(рис. 3.17)

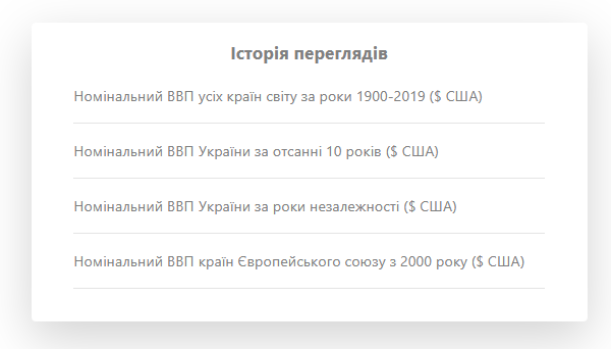

Рис. 3.17 – Блок з історією попередніх переглядів користувача

Для пошуку даних користувач повинен ввести в пошукове поле пошуковий запит. Це може бути як слово, так і словосполучення, чи синонім до слова. Після введення запиту користувач потратить на сторінку з результатами пошуку (рис. 3.18).

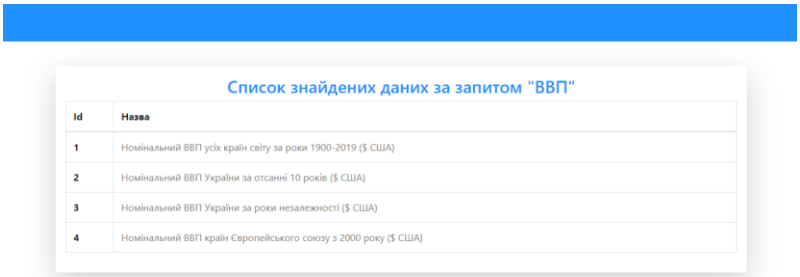

Рис. 3.18 – Сторінка результатів пошуку даних

Користувач повинен обрати необхідні йому дані та клікнути по ним. Після цього користувач потрапить на наступну сторінку, безпосередньо з самими статистичними даними (рис. 3.19а – 3.21).

Рис. 3.19а – Графічне представлення статистичних даних у вигляді графіку

Рис. 3.19б – Графічне представлення статистичних даних у вигляді стовпчикової діаграми

На даній сторінці користувач може:

1) Проглянути сумарну статистику(рис. 3.19а).
2) Зробити вибірку статистики за країною чи роком за допомогою блоку зліва (рис. 3.20).

3) Завантажити статистичні дані на свій гаджет у трьох різних форматах (рис. 3.21).

Рис. 3.20 – Блок для графічного представлення вибірки статистичних даних

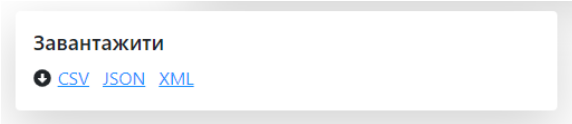

Рис. 3.21 – Блок для завантаження статистичних даних

Висновки до розділу

1. Була реалізована модель бази даних на основі DDF.
2. Було реалізовано бізнес-логіку додатку у серверній частині системи.
3. Було протестовано систему як мануально(перевірка на валідність усіх текстових полів), так і програмно(JUnit тести).
4. Було реалізовано інструкцію користувача для подальшої роботи користувача з системою.

Отже, веб-сервіс для провайдера статистичних даних був реалізований за допомогою платформи Spring Framework та мови програмування Java для серверної частини. Для роботи з базою даних вибрано Spring Data та ORM Hibernate. Для оформлення представлення (View) обрано фреймворк ReactJS та мову програмування JavaScript. Тестування проводилося за допомогою бібліотеки JUnit, результати тестування були успішні.

ВИСНОВКИ

У ході виконання дипломної роботи спроектована та розроблена система для провайдера статистичних даних. Система являє собою веб-додаток для адміністрації сервісу та користувачів, які хочуть проаналізувати чи надати певні статистичні дані. Додаток спрощує роботу в управлінні статистичними даними, бо всі дані які записуються в БД уніфіковані, що дозволяє завантажувати майже будь-яку статистику з різних джерел до додатку. Також, тепер аналітикам та звичайним людям не потрібно аналізувати статистику на папері.

У процесі аналізу готових рішень визначено їх основні переваги та недоліки і відповідно спроектовано цільову систему. Проведений аналіз дозволив зрозуміти основні складнощі при побудові даної системи, що системи такої складності є досить важкі в проектуванні і тому в більшості випадках мають закриту архітектуру. Окрім того, виявлено, що майже всі

програмні додатки для статистичними даними не мають функціонального графічного інтерфейсу та можливості завантаження статистики у табличних форматах. Дана система буде абсолютно безкоштовною, з відкритою архітектурою та з часом буде розширюватись.

Отримані знання та навички являють дуже корисними для моєї професійної діяльності. При написанні даної дипломної роботи закріплено навички зі створення програмних продуктів за допомогою платформи Spring Framework, мови Java та модулей Spring Security, Spring Data, Spring MVC, фреймворка ReactJS та інших.

ПЕРЕЛІК ПОСИЛАНЬ

1. WorldBank. [Електронний ресурс] – Режим доступу: http://www.data.worldbank.org
2. CitiBike. [Електронний ресурс] – Режим доступу: https://www.citibike.com
3. GovUa. [Електронний ресурс] – Режим доступу: http://www.data.gov.ua
4. Вікіпедія. Триярусна архітектура [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Триярусна_архітектура
5. Habr. Модуль Spring Security. [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/203318/
6. Habr. Модуль Spring Data. [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/435114/

7. Habr. Модуль Spring MVC. [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/336816/

8. Сучасна статистика. [Електронний ресурс] – Режим доступу: https://academy.gov.ua/ej/ej3/txts/socialna_statistika/07-ostapchuk.pdf

9. Вікіпедія. Статистика [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Статистика

10. Мейер М. Теория реляционных баз данных. – М.: Мир, 1987 – 608с.

11. Вікіпедія. Нормальная форма [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Нормальна_форма

12. Вікіпедія. Модель «сутність – зв'язок» [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Модель_сутність_–_зв%27язок

13. Воробьев С.В. Начало работы с базами данных: Учебно-методическое пособие. -М.: МИЭМ, 2008. - 60 с.

14. Архітектура REST. [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/mailru/blog/345184/